Ingeniería de Software

Introducción

La realidad que todos los días vivimos, lo que nos rodea, nuestra vida cotidiana está cada día más en contacto con el software, desde nuestra actividad bancaria, nuestra salud, nuestras tareas cotidianas, hasta en nuestra vida hogareña... Vale decir que desde las cosa más sencillas como una tostadora hasta sistemas crítico como aquellos de los que pueda depender nuestra salud y por ende nuestra vida, están directamente conectados con el software.

Es por esta razón que la Ingeniería de Software va cobrando día a día mayor relevancia en todo el mundo.

Las "*buenas prácticas*" de la Ingeniería de Software deben asegurar que el software facilite nuestras vidas, cada vez más ligadas a la Informática.

Qué es, entonces la Ingeniería de Software, es algo que debemos definir con la mayor precisión posible a fin de establecer las bases de lo que se estudiará.

Para ese fin, hemos establecido, una serie de definiciones sobre las que se ha trabajado en conjunto, llegando a la conclusión de que muchos términos se utilizan con significados parecidos, que otros son utilizados erróneamente y que en síntesis, al ser tanto la Informática como la Ingeniería de Software dos disciplinas jóvenes, coexisten estos problemas conceptuales que sólo agregan confusión a la creación y producción de software de calidad.

Definiciones

Ingeniería de Software

Ingeniería de Software es el estudio de los *principios y metodologías* para el desarrollo y mantenimiento de sistemas de software [Zelkovitz, 1978]

Ingeniería del Software es la aplicación práctica del conocimiento científico en el diseño y construcción de programas de computadoras y la documentación asociada requerida para desarrollar, operar y mantenerlos. Se conoce también como desarrollo de software o producción de software [Bohem, 1976]

Ingeniería del Software trata del establecimiento de los *principios y métodos de la ingeniería* a fin de obtener software de modo rentable, que sea *fiable* y trabaje en máquinas reales [Bauer, 1972]

La aplicación de *un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación* (funcionamiento) y mantenimiento del software, es decir, la aplicación de ingeniería al software [IEEE, 1993]

Método

Según el diccionario:

Método

(gr. *méthodos* ← *hodós,* camino)
substantivo masc
1 modo ordenado de proceder para llegar a un resultado o fin determinado, esp. para descubrir la verdad y sistematizar los conocimientos.

Para la disciplina

Los métodos de la Ingeniería de Software indican cómo construir software. Incluyen una gran gama de tareas como el análisis de requisitos, diseño, elaboración de programas, pruebas y mantenimiento.

Herramientas

Según el diccionario:

Herramienta

(lat. *ferramenta,* pl)
substantivo fem

1 instrumento, generalmente. de hierro, con que trabajan los artesanos.
2 conjunto de estos instrumentos.

Para la disciplina:

Las *herramientas* de la Ingeniería de Software indican proporcionan un soporte para el proceso y para los métodos. Pueden ser automáticas y o semiautomáticas.

- *Herramientas CASE* (Computer-Aided Software Engineering)

Son instrumentos que permiten realizar algo con mayor exactitud o de mejor forma. Nos ayudan a ser más productivos o eficientes o refuerzan la calidad del producto resultante

Proceso

Según el diccionario:

Procedimiento

substantivo masc

1 acción de proceder.
2 método de ejecutar algunas cosas.

Para la disciplina

Proceso en Ingeniería de Software define el marco de trabajo para un conjunto de las actividades del desarrollo del software y lo abarca en totalidad.

Paradigma

Según el diccionario

Paradigma

(gr. *parádeigma,* modelo)

substantivo masc

1 Ejemplo que sirve de norma, esp. de una conjugación o declinación.

2 LING conjunto virtual de elementos de una misma clase gramatical, que pueden aparecer en un mismo contexto.

Para la disciplina

Paradigma: representa un enfoque particular o filosofia para la construcción de Software, como el desarrollo orientado a objetos.

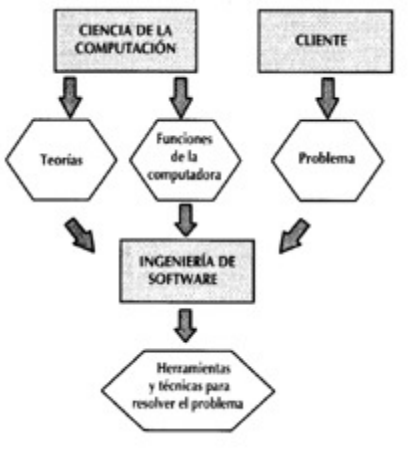
La clarificación de estos conceptos no sólo brindan un marco más claro para el desarrollo de la disciplina sino que también permiten mejorar la comunicación entre profesionales y con los clientes, usuarios, etc.

Otras definiciones que usamos con frecuencia y en forma poco clara son términos como Informática, Ciencias de la computación e Ingeniería de Software inclusive.

Para aclarar estos términos, diremos que a nivel internacional se estila actualmente, llamar "ciencias de la computación", traducción del inglés, "Computer Science" como sinónimo de Informática en general. Ese término no sólo nos lleva a pensar en software sino también en la computadora como objeto de estudio y de investigación, lo que se refiere al hardware, no siendo éste el caso de la Ingeniería de Software.

El ingeniero de software utiliza a las computadoras como una herramienta, como parte de la solución que se brinda a la resolución de una problemática, pero son sólo una herramienta.

Pfleeger[1] ha descrito la ubicación de la Ingeniería de Software dentro de la Ciencia de la Computación con el siguiente gráfico:



[1] [Pfleeger, 2001]

Esta visión si bien controvertida pero en constante evolución, no es acordada por todos.

Los ingenieros de software usan herramientas, técnicas, procedimientos y paradigmas eficientes y productivos para generar soluciones efectivas a los problemas.

Una cosa sí parece estar clara: *la Ingeniería de Software es una disciplina de diseño y desarrollo de software de alta calidad.*

Esta aseveración podría debatirse largamente, implicando temas controvertidos como qué es la calidad, las diferentes visiones de la misma, cómo medirla y cómo producir software de calidad dentro del marco de una economía globalizada y con un avance de la información y la tecnología altamente acelerados.

Lo que sí es seguro es que la improvisación es el camino menos conveniente si queremos que una empresa de software trabaje de forma seria y con cierta proyección en el tiempo y en el mercado.

Más avanzados en la materia se verán con más profundidad aspectos de la calidad: criterios, factores, métricas y procedimientos que aseguren la calidad en los productos software.

Sin embargo, se deberá tener en cuenta que el aseguramiento de la calidad es una de las tareas más importantes en la Ingeniería de Software; producir software confiable e íntegro teniendo como base criterios amplios y que tiendan a una planificación y proyección que vaya más allá de lo inmediato, teniendo en cuenta parámetros como la confiabilidad, los posibles usos indebidos, etc.

Lamentablemente, la calidad, muchas veces se ve muchas veces restringida por las exigencias del mercado que hace que un producto salga a la venta o se implante sin el tiempo de hacer las pruebas pertinentes, en forma segura y eficaz, hace que sea común que las primeras versiones de algunos productos sean versiones sin una calidad respetable y sólo a través de la aparición de versiones sucesivas se vayan corrigiendo, errores que no debieran haber existido. Otra de las causas, es la falta de preparación en el equipo de pruebas que suele estar preparado para testear sólo las funciones más frecuentemente utilizadas.

La calidad debe ser una premisa que acompañe todo el proceso software, vale decir que se deberá realizar tareas de aseguramiento de la calidad a lo largo de todo el ciclo de vida del producto software, y es uno de los objetivos de la Ingeniería de Software.

La ausencia de calidad puede ser costosa si no se aplica al proceso software. Los errores son más costosos conforme se detecten en las etapas más tardías del desarrollo.

El costo de corrección de un error en la fase de análisis se estima en una décima parte de lo que cuesta corregir un error similar una vez que el software se entregó al cliente.

Dos técnicas muy simples y poderosas de detección de errores son las revisiones y las inspecciones, técnicas que veremos con se realizan más adelante, han probado detectar 4 de cada 5 errores introducidos en el proceso de construcción del software. Obviamente, se habla siempre de la calidad aplicada a todo el proceso software y no sólo a la fase de codificación.

La IEEE ha dedicado grandes esfuerzos a la elaboración de normas para el aseguramiento de la calidad.

La calidad debe hacerse desde tres perspectivas:

- ✓ Producto
- ✓ Proceso
- ✓ Ámbito del negocio o dominio

La calidad del producto es juzgada por usuarios, diseñadores, programadores y aquellos que realizan el mantenimiento. A partir de esta idea, y como resulta obvio de pensar, la idea de calidad, evidentemente será diferente ya que son muy diferentes, las visiones del software que cada uno de ellos.

El usuario pensará que el software es de gran calidad si hace lo que se espera de él, fácil de aprender, etc. Por otra parte, quienes diseñan y escriben y para aquellos que deben mantener juzgarán las características internas de los productos.

Desde el punto de vista del proceso, la utilización de acciones que contribuyen a la calidad, no deberán ser descuidadas ya que un proceso de calidad, asegura un producto de calidad. Considerándose de vital importancia la garantía de calidad en el proceso, existen normas que califican a las empresas en diferentes grados de madurez según las actividades de calidad que se realizan a lo largo de todo el proceso software (CMM, SPICE, ISO9000)

Cambios en la Ingeniería de Software

Las tradicionales actividades del desarrollo del software: Análisis y definición de requisitos, Diseño del sistema, Diseño de programas, Prueba unitaria, Prueba de integración, Prueba de Sistema, Entrega del Sistema y mantenimiento, en un modo ideal, parecen sugerir una secuencia lógica, que en la realidad no siempre es posible. En muchos casos las actividades se repiten para ajustarlas, por lo que en el *proceso de desarrollo software* estas nueve actividades se organizan de diferentes formas para lograr un mejor resultado, dadas las características de un determinado proyecto. Los proyectos alcanzan mayor tamaño día a día, crece la complejidad en cuanto a la interacción entre diferentes tipos de software, protocolos de comunicación, redes de gran alcance, sistemas embebidos, etc. Todos estos vertiginosos cambios han provocado también cambios en la Ingeniería de Software que busca adaptarse como disciplina a las necesidades del desarrollo de software.

Wasserman, en su libro *"Towards a discipline of software engineering"*, analiza los cambios en la Ingeniería de Software, definiendo las causas en 7 factores que los originan:

1. Criticidad del tiempo de puesta en el mercado para los productos comerciales .
2. Variaciones en los factores económicos de la computación, menores costos del hardware y mayores costos del desarrollo y el mantenimiento.
3. Disponibilidad de la poderosa computación de escritorio.
4. Uso intensivo de redes locales y de área amplia.
5. Disponibilidad y adopción de tecnología orientada a objetos.
6. Interfaces gráficas de usuario, usando ventanas, iconos y punteros.
7. La falta de predicción del modelo de desarrollo de software en cascada.

Otro aporte de Wasserman son las ocho *nociones fundamentales* que forman las bases para una disciplina efectiva de la Ingeniería de Software. Ellas son:

1. ***Abstracción*** Permite realizar una descripción del problema desde un nivel de generalización que permita concentrarse en los conceptos fundamentales.

   La abstracción se permite realizar diferentes niveles de conceptualización del problema.

2. ***Métodos y notaciones de análisis y diseño*** No existe en Ingeniería de Software una notación que permita la comunicación entre pares. Esta situación conlleva malas interpretaciones constituyéndose en unos de los problemas claves de la Ingeniería de Software. Los métodos de análisis y diseño, aportan además de un medio de comunicación, la posibilidad de construir modelos y probarlos en cuanto a completitud y consistencia.

3. ***Prototipado de la interfaz de usuario*** Esta técnica permite construir una versión reducida de un sistema, una funcionalidad limitada, que puede usarse para ayudar al usuario o al cliente a identificar los requerimientos del sistema, o para demostrar la factibilidad de un diseño o enfoque determinado. También puede favorecer aspectos o suposiciones no aclarados en la educción de requisitos.

4. ***Arquitectura del software*** La arquitectura es una noción sumamente importante en todo el proceso software ya que afectan no sólo a la facilidad de implementación, prueba, velocidad, efectividad de mantenimiento y el cambio. La calidad de la arquitectura es definitiva en un sistema. Algunos autores, sostienen que la arquitectura puede constituirse como una disciplina en sí misma, que persigue como fin último la división en módulos arquitectónicos lo más independientes posible de otros.

5. ***El proceso del software*** La organización y la disciplina en las actividades han sido reconocidas por contribuir a la calidad del software resultante. Sin embargo, diferentes productos necesitan de diferentes procesos y no existe una receta que cubra todas las necesidades de todos los desarrollos.

6. ***Reutilización*** Permite sacar ventaja de los aspectos comunes de las aplicaciones, reutilizando elementos de desarrollos previos. Se pueden reutilizar conjuntos de requerimientos, partes de diseños o grupos de guiones de prueba. Si bien la reutilización es un objetivo claro para las empresas de desarrollo de software, no es fácil su implementación debido a muchos factores como tiempo de desarrollo de módulos reutilizables, difícil documentación, difícil de establecer el grado de calidad del módulo con la calidad exigida para un proyecto, etc.

7. ***Medidas*** A través de las diferentes áreas de investigación que intentan mejorar la Ingeniería de Software un aspecto clave para la mejora ha sido la medición, la que permite establecer dónde se está y hacia dónde se pretende ir, constituyendo una de las “buenas prácticas” de la Ingeniería de Software, ya que permite comparar, evaluar y retroalimentar el proceso software.

8. ***Herramientas y ambientes integrados*** Este factor hace referencia a la utilización de herramientas *CASE* las que favorecen un ambiente integrado y estandarizado que mejora el proceso de desarrollo, si bien, la confluencia de dos características del desarrollo hacen difícil esta integración: por un lado los problemas son diferentes y requieren de diferentes enfoques en el proceso software, por lo que el uso de diferentes procesos, métodos y re-

cursos hacen difícil la unificación del enfoque. Sin embargo, la existencia de herramientas cada vez mejores, permiten analizar y concluir cuál es la más apropiada para un problema o una aplicación. <revisar>.

El Proceso Software y los Modelos de Ciclo de Vida

Elegir un *Ciclo de vida*, permite encadenar las tareas y actividades del *Proceso Software*. Una vez conseguido un proceso software concreto para un proyecto determinado se está preparado para planificar los plazos del proyecto, asignar personas a las diferentes tareas, presupuestar costos del proyecto, etc.

Los *procesos software,* hoy día se usan como:

- ✓ Guía descriptiva sobre la documentación a producir y enviar al cliente
- ✓ Bases para determinar qué herramientas, técnicas y metologías de Ingeniería de Software serán más apropiadas para brindar soporte a las diferentes actividades.
- ✓ Marco para analizar o estimar patrones de asignación y consumo de recursos a lo largo de todo el ciclo de vida del producto software.
- ✓ Base para llevar estudios empíricos para determinar qué afecta a la productividad, costo y calidad del software.
- ✓ Descripción comparativa sobre cómo los sistemas llegan a ser lo que son.

El proceso software está compuesto de cuatro procesos principales:

1. Proceso de selección de un modelo de ciclo de vida
2. Proceso de gestión del proyecto
3. Procesos orientados del desarrollo del software
4. Procesos integrales del proyecto

Estos procesos abarcarán todo el ciclo de vida del producto desde su comienzo hasta su retiro por obsolescencia o por reemplazo, incluyendo dentro del mismo, el mantenimiento.

Material de lectura sugerido: Modelos de ciclos de vida de Juristo, Pfleeger y Pressman

Proceso software.pdf